Botones e Iconos

1. Añadir comentarios a la vista del componente DetalleExcursion

En primer lugar, se crea un nuevo *Card* para renderizar los comentarios asociados al detalle de cada una de las excursiones. Para conseguir esto, se crea una nueva clase funcional de nombre *renderComentarios(props)* dentro del fichero *DetalleExcursionComponent.js*, de manera análoga a lo que hicimos con la clase funcional *renderExcursion(props)*. Se introduce el FlatList dentro de un Card, muy similar a lo realizado en el commit anterior en QuienesSomos:

```
function RenderComentario(props) {

  const comentarios = props.comentarios;

  const renderCommentarioItem = ({item, index}) => {

      return (
          <View key={index} style={{margin: 10}}>
              <Text style={{fontSize: 14}}>{item.comentario}</Text>
              <Text style={{fontSize: 12}}>{item.valoracion} Stars</Text>
              <Text style={{fontSize: 12}}>{'-- ' + item.autor + ', ' + item.dia} </Text>
</View>
      );
  };

  return (
    <Card>
      <Card.Title>Comentarios</Card.Title>
      <Card.Divider/>
      <FlatList
          data={comentarios}
          renderItem={renderCommentarioItem}
```

La información a mostrar será obtenida del fichero JavaScript *comentarios.js* de la carpeta *comun*.
Como resultado se obtiene lo siguiente:

2. Botones o Iconos

A continuación, se añade el botón para indicar como favorita una excursión, empleando el componente *Icon* de la librería React Native Elements. Para cambiar la apariencia del botón será necesario saber cuándo ha sido clicado.

De momento el estado del botón solo se mantendrá mientras la página no se refresque y no se salga del fichero DetallesExcursion.js, más adelante utilizando Redux se podrá mantener aunque la página se refresque y se salga de este archivo js.

Dentro de la clase funcional RenderExcursion se inserta el código con el elemento Icon:

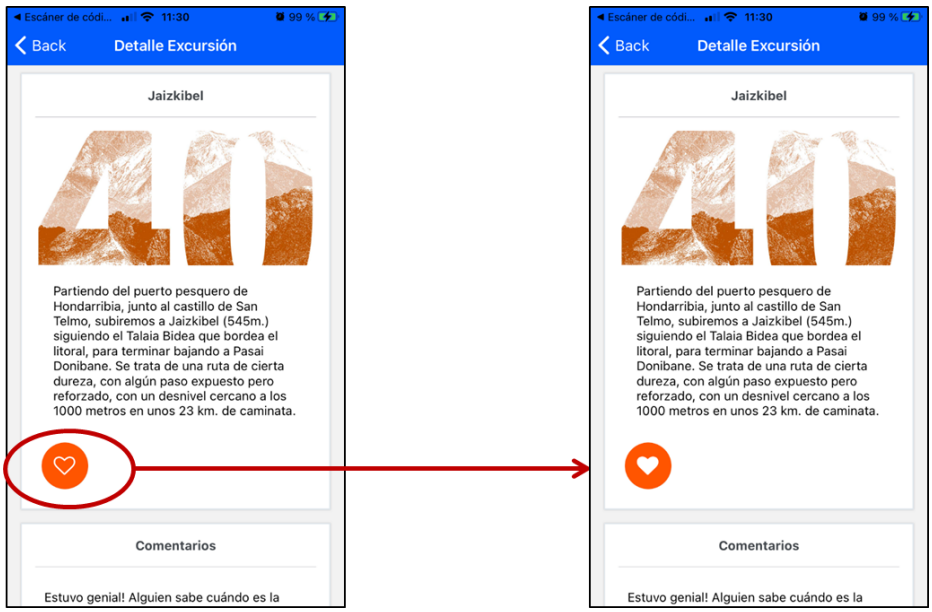
```
<Icon
    raised
    reverse
    name={ props.favorita ? 'heart' : 'heart-o'}
    type='font-awesome'
    color='#f50'
    onPress={() => props.favorita ? console.log('La excursión ya se encuentra entre las favoritas') : p
/>
```

Se observa como en el name si se le pasa como props favorita representa un icono u otro (equivale a un if-else), al igual que en onPress que al pulsarlo muestra en la consola que la excursión ya está entre las favoritas.

Se terminar de editar el archivo con el código proporcionado, cabe tambié destacar:

- La utilización de some que comprueba si al menos un elemento del array cumple con la condición implementada por la función proporcionada. También se utiliza el método filter que crea un nuevo array con todos los elementos que cumplan la condición implementada por la función dada.

Como resultado se obtiene lo siguiente:



3. Customizar el Drawer Menu

Lo primero, instalamos la siguiente librería para garantizar una correcta renderización de los márgenes de seguridad en nuestra aplicación. En mi caso tuve que instalar la versión 3.1.9, ya que sino me resultaba incompatible:

yarn add react-native-safe-area-context@v3.1.9

Se edita el DrawerMenu para hacerlo más atractivo añadiendo iconos y el logo en la parte superior.

Se añade el componente funcional CustomDrawerContent, que introduce el logo en la parte superior del menú:

```
function CustomDrawerContent(props) {
  return (
    <DrawerContentScrollView {...props}>
      <SafeAreaView style={styles.container} forceInset={{ top: 'always', horizontal: 'never' }}>
        <View style={styles.drawerHeader}>
          <View style={{flex:1}}>
          <Image source={require('./imagenes/logo.png')} style={styles.drawerImage} />
          </View>
          <View style={{flex: 2}}>
            <Text style={styles.drawerHeaderText}> Gaztaroa</Text>
          </View>
        </View>
        <DrawerItemList {...props} />
      </SafeAreaView>
    </DrawerContentScrollView>
  );
}
```

Dentro de cada Drawer.Screen se añade el siguiente código para añadir los iconos a la izquierda, en función del icono deseado el campo name variará:

```
options={{
  drawerIcon: ({ tintColor }) => (
    <Icon
    name='home'
    type='font-awesome'
    size={22}
    color={tintColor}
    />
  )
  }}
```

Se crea también un Style.Sheet para dar formato al componente funcional CustomDrawerContent.

Además de esto, se desea añadir un botón en la parte superior izquierda de cada una de las pantallas, cuya función será desplegar el *Drawer Menu*. Para ello, haremos uso de un icono, en combinación con la propiedad *headerLeft* y el método *DrawerActions.toggleDrawer()* como se muestra en el siguiente código:

```
function HomeNavegador({ navigation }) {
  return (
    <Stack.Navigator
      initialRouteName="Home"
      headerMode="screen"
      screenOptions={{
        headerTintColor: '#fff',
        headerStyle: { backgroundColor: '#015afc' },
        headerTitleStyle: { color: '#fff' },
        headerLeft: () => (<Icon name="menu" size={28} color= 'white' onPress={ () => navigation.dispatch(DrawerActio
      }}>
```

Será necesario en todas las partes menos en Calendario que se añadirá este código al Stack.Screen de Calendario y no a Detalles ya que este mantendrá la opción de Back.
Como resultado final se obtiene lo siguiente: